

### **REMARKS/ARGUMENTS**

This paper is being provided in response to the Office Action dated December 20, 2006 for the above-referenced application. In this response, Applicant has cancelled Claims 2, 5, 18, 21, 23, 26, 39 and 42, and amended Claims 1, 3, 4, 6, 8, 11, 12, 16, 17, 19, 20, 22, 24, 25, and 27-41. Applicant respectfully submits that the amendments to the claims and the newly added claims are supported by the originally filed application.

The rejection of Claims 1, 20, 22, and 41 under the non-statutory obviousness-type double patenting as being unpatentable over Claims 8 and 35 of U.S. Patent No. 6,701,519 ('519 patent) is hereby traversed and reconsideration thereof is respectfully requested in view of amendments and remarks herein.

Applicant's amended Claim 1 recites, in part, *a computer implemented method for automatically tracking build information comprising: extracting build information by processing, for each of one or more builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said one or more builds;...*

Claim 8 of the '519 patent depends, directly or indirectly, from Claims 1, 2, and 7. Applicant respectfully submits that Applicant's amended Claim 1 is patentably distinct from Claim 8 of the '519 patent, and any claims that Claim 8 depends from, in that at least the

foregoing recited feature of Applicant's amended Claim 1 does not have an equivalent corresponding step included therein and is also not obvious in view of the foregoing claims of the '519 patent.

In connection with the '519 patent, Claim 35 recites features similar to those of Claim 8 and depends, directly or indirectly, from Claims 29, 30, and 34. Claims 29, 30 and 34 of the '519 patent recite features similar those of Claims 1, 2 and 7 of the '519 patent. Thus Applicant's Claim 1 is patentably distinct over Claim 35, and claims that Claim 35 depends from, for reasons similar to those set forth above.

Applicant's amended Claims 20, 22, and 41 recite features similar to those of Claim 1. Thus, Applicant's Claims 20, 22, and 41 are also patentably distinct over the claims of the '519 patent for reasons similar to those set forth above regarding Applicant's independent Claim 1.

Applicant respectfully submits that the claims in the instant application may be further amended in connection with subsequent prosecution. As such, in the event that the Examiner still maintains this rejection, Applicant will consider filing a terminal disclaimer when there are no other rejections and objections and the case is otherwise in condition for allowance.

In view of the foregoing, Applicant respectfully requests that the rejection be reconsidered and withdrawn.

The rejection of Claims 1-2, 4, 6-23, 25, 27-42 under 35 U.S.C. 101 as being directed to non-statutory subject matter is hereby traversed and reconsideration thereof is respectfully

requested. This rejection as applied to Claims 2, 18, 21, 23, 39, and 42 is moot in view of the cancellation of these claims herein.

Applicant's amended Claim 1 recites a computer implemented method for automatically tracking build information. Build information is extracted by processing, for one or more builds, one or more software modules produced using a compilation process. One or more builds are registered by storing build information in a database. Software module information is automatically determined about software being tested. A first of the builds included in the database is determined in which the build corresponds to the software module information.

Applicant respectfully submits that Claim 1, and claims that depend therefrom, are directed to statutory subject matter. As pointed out above, Applicant's amended Claim 1 is not directed to an abstract, non-practical application. Applicant's Claim 1 is directed to a computer implemented method which includes storing information in a database and using the information in the database to determine a previously registered build that corresponds to software module information of software being tested.

Applicant's independent Claim 22, and claims that depend therefrom, are directed to a computer readable medium with comprising executable code thereon. Claim 22 recites features similar to those of Claim 1. Applicant respectfully submits that Claim 22 is directed to statutory subject matter for reasons similar to those set forth above regarding Claim 1. Additionally, Claim 22 now is directed to a computer readable medium which is statutory subject matter under 35 U.S.C. 101.

Applicant's amended Claim 16 recites a computer implemented method for determining a code volatility metric. Build information is extracted by processing, for one or more builds, one or more software modules produced using a compilation process. One or more builds are registered by storing build information in a database. A first and a second of the registered builds are identified by retrieving information from the database. A query of the database is performed to determine code volatility between software modules included in both the first and second builds. The code volatility metric is calculated using information from the database. The code volatility metric is determined using one or more metrics representing an amount of code change between software modules in both the first and second builds.

Applicant's independent Claim 16, and claims that depend therefrom, are directed to statutory subject matter. Applicant's amended Claim 16 is not directed to an abstract, non-practical application. As pointed out above, Applicant's Claim 16 is directed to a computer implemented method which includes storing information in a database, retrieving information from the database, performing a query of the database and using the information from the database to determine a code volatility metric.

Applicant's independent Claim 37, and claims that depend therefrom, are directed to a computer readable medium with comprising executable code thereon. Claim 37 recites features similar to those of Claim 16. Applicant respectfully submits that Claim 37 is directed to statutory subject matter for reasons similar to those set forth above regarding Claim 16. Additionally, Claim 37 now is directed to a computer readable medium which is statutory subject matter under 35 U.S.C. 101.

Applicant's amended Claim 20 recites a computer implemented method for automatically tracking build information. Build information is extracted by processing, for one or more builds, one or more software modules produced using a compilation process. One or more builds are registered by storing build information in a database. A software program is executed. Executing the software program includes processing software modules included in the software program during execution of the software modules to determine module information. Using build information from the database and the module information, a matching build of the previously registered builds is determined. Testing information associated with the matching build is determined by performing a query of the database.

Applicant respectfully submits that Claim 20 is directed to statutory subject matter. As pointed out above, Applicant's amended Claim 20 is not directed to an abstract, non-practical application. Applicant's Claim 20 is directed to a computer implemented method which includes storing information in a database, using the information in the database, and querying the database.

Applicant's independent Claim 41 is directed to a computer readable medium with comprising executable code thereon. Claim 41 recites features similar to those of Claim 20. Applicant respectfully submits that Claim 41 is directed to statutory subject matter for reasons similar to those set forth above regarding Claim 20. Additionally, Claim 41 now is directed to a computer readable medium which is statutory subject matter under 35 U.S.C. 101.

For at least the foregoing, reasons, Applicant respectfully submits that the claims are now directed to statutory subject matter.

In view of the foregoing, Applicant respectfully requests that the rejection be reconsidered and withdrawn.

The rejection of Claims 20-21 and 41-42 under 35 U.S.C. 112, second paragraph, is hereby traversed and reconsideration thereof is respectfully requested. The rejection as applied to Claims 21 and 42 is moot in view of the cancellation of these claims herein. Applicant has amended Claim 20 and 41 in accordance with remarks set forth in the Office Action.

In view of the foregoing, Applicant respectfully requests that the rejection be reconsidered and withdrawn.

The rejection of Claims 1-10, 16-31, and 37-42 under 35 U.S.C. 102(b) as being anticipated by Leblang (U.S. Patent No. 5,574,898, hereinafter “Leblang”) is hereby traversed and reconsideration thereof is respectfully requested. This rejection as applied to Claims 2, 5, 18, 21, 23, 26, 39 and 42 is moot in light of the cancellation of these claims herein.

Applicant’s amended Claim 1 recites a computer implemented method for automatically tracking build information comprising: extracting build information by processing, for each of one or more builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output

of a compilation process for each of said one or more builds; registering said one or more builds by storing said build information corresponding to each of said one or more builds in a database; automatically determining software module information about software being tested, wherein said software module information is gathered from one or more software modules during execution of said software being tested; and automatically determining a first of said one or more builds included in the database which corresponds to said software module information. Claims 3-4, 6-10 depend from Claim 1.

Applicant's amended Claim 16 recites a computer implemented method for determining a code volatility metric, the method comprising: extracting build information by processing, for at least two builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said at least two builds; registering said at least two builds by storing said build information corresponding to each of said at least two builds in a database; identifying, by retrieving at least a portion of said build information from the database, a first and a second of said at least two builds; performing a query of the database to determine code volatility between software modules included in both the first and the second of said at least two builds; and calculating, in response to said query, said code volatility metric using said build information including software module information about said first and said second builds included in the database, said code volatility metric being determined using one or more metrics representing an amount of code change that has occurred between software modules in both said first build and said second build. Claims 17 and 19 depend therefrom.

Applicant's amended Claim 20 recites a computer implemented method for tracking build information comprising: extracting build information by processing, for each of one or more builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said one or more builds; registering said one or more builds by storing said build information in a database; executing a software program that includes one or more software modules, said executing including processing said one or more software modules of the software program during execution of said software program to determine module information about said one or more software modules of said software program; automatically determining, using said build information included in the database and said module information obtained from said executing, a matching build for said one or more software modules of the software program that are also associated with one of said builds previously registered; and determining testing information associated with the matching build by performing a query of said database, said testing information including at least one type of runtime analysis performed for the matching build.

Applicant's amended Claim 22 recites a computer readable medium comprising machine executable code stored thereon for automatically tracking build information, the computer readable medium comprising: machine executable code for extracting build information by processing, for each of one or more builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build



information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said one or more builds; machine executable code for registering said one or more builds by storing said build information corresponding to each of said one or more builds in a database; machine executable code for automatically determining software module information about software being tested, wherein said software module information is gathered from one or more software modules during execution of said software being tested; and machine executable code for determining a first of said one or more builds included in the database which corresponds to said software module information. Claims 24, 25, 27-31 depend therefrom.

Applicant's amended Claim 37 recites a computer readable medium comprising machine executable code stored thereon for determining a code volatility metric, the computer readable medium comprising: machine executable code for extracting build information by processing, for at least two builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said at least two builds; machine executable code for registering said at least two builds by storing said build information corresponding to each of said at least two builds in a database; machine executable code for identifying, by retrieving at least a portion of said build information from the database, a first and a second of said at least two builds; machine executable code for performing a query of the database to determine code volatility between software modules included in both the first and the second of said at least two builds; and machine executable code for calculating, in response to said query, said code

volatility metric using said build information including software module information about said first and said second builds included in the database, said code volatility metric being determined using one or more metrics representing an amount of code change that has occurred between software modules in both said first build and said second build. Claims 38 and 40 depend therefrom.

Applicant's amended Claim 41 recites a computer readable medium comprising executable code stored thereon for tracking build information, the computer readable medium comprising: machine executable code for extracting build information by processing, for each of one or more builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said one or more builds; machine executable code for registering said one or more builds by storing said build information in a database; machine executable code for executing a software program that includes one or more software modules, said executing including processing said one or more software modules of the software program during execution of said software program to determine module information about said one or more software modules of said software program; machine executable code for automatically determining, using build information included in the database and said module information obtained from said executing, a matching build for said one or more software modules of the software program that are also associated with one of said builds previously registered; and machine executable code for determining testing information associated with the matching build

by performing a query of said database, said testing information including at least one type of runtime analysis performed for the matching build.

Leblang's Figure 1 includes a version control system and stores of source objects and derived objects. Derived objects are created by running a system build process on a particular version of the source objects (See Figure 1, Col. 5, Lines 36-45). In Figure 7, the version control system is illustrated for use with two developers each using a different view having a set of versions from the versioned object bases (VOBs). (See Figures 2, 7; Col. 6, Lines 8-13; Col. 8, Line 51-Col. 10, Line 9). Source files are the input to the software build process. Equally important are the files that are created by software builds. With the version control system, such files are called derived objects. Each derived object has two main parts, the data itself and an associated configuration record. The configuration record is a "bill of materials" that stores an audit of the build that produced the derived object. This automatically includes a list of the element versions used in the build. It also includes versions of dependencies (such as build tools) that are explicitly declared in the makefile. Derived objects, like elements, can be accessed either with standard UNIX pathnames or with version extended names. (Col. 25, Lines 27-41).

Applicant's Claim 1, as amended herein, is neither disclosed nor suggested by Leblang in that Leblang neither discloses nor suggests as least the features of ***a computer implemented method for automatically tracking build information comprising: extracting build information by processing, for each of one or more builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software***

*module produced as an output of a compilation process for each of said one or more builds;  
... automatically determining software module information about software being tested,  
wherein said software module information is gathered from one or more software modules  
during execution of said software being tested ... as set forth in Claim 1.*

Leblang's version control system is used to maintain particular versions of source objects 24 used in producing builds. As described above, Leblang discloses a configuration record that includes a list of element versions used in the build and dependencies explicitly declared in the makefile. Leblang appears to use makefiles to determine the configuration record. Makefiles are used to specify build scripts with different inputs (e.g., source files) and outputs (e.g., binaries) generated. However, Leblang does not disclose or fairly suggest extracting build information by processing software modules produced using a compilation process resulting in the software modules. Rather, Leblang operates on makefiles to determine the configuration record and appears silent regarding extracting any information from binaries, and the like, that may be produced using a compilation process. As such, Leblang neither discloses nor fairly suggests *extracting build information by processing, for each of one or more builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said one or more builds*, as set forth in Claim 1.

Additionally, Leblang discloses execution of build scripts but appears silent regarding gathering software module information about software being tested during execution of software

being tested. As such, Leblang neither discloses nor fairly suggests ... *automatically determining software module information about software being tested, wherein said software module information is gathered from one or more software modules during execution of said software being tested* ... as set forth in Claim 1.

For at least these reasons, Claim 1, and claims that depend therefrom, are neither disclosed nor suggested by Leblang. Claims 20, 22, and 41 recite features similar to those of Claim 1. Thus, Claims 20, 22 and 41, and claims that depend therefrom, are neither disclosed nor suggested by Leblang for reasons similar to those set forth regarding Claim 1.

Claim 16 recites an extracting step similar to that as set forth in Claim 1 which is neither disclosed nor suggested by Leblang. Claim 16 also recites an additional step of calculating a code volatility metric which is also neither disclosed nor suggested by Leblang. In particular, Leblang neither discloses nor suggests *calculating, in response to said query, said code volatility metric using said build information including software module information about said first and said second builds included in the database, said code volatility metric being determined using one or more metrics representing an amount of code change that has occurred between software modules in both said first build and said second build*, as set forth in Claim 16. Leblang's system is used to maintain and track different versions of sources (See, for example, Figures 5 and 7). However, Leblang appears to make no disclosure or suggestion of calculating a code volatility metric *representing an amount of code change that has occurred between software modules in both said first build and said second build*. As such,

Leblang also neither discloses nor fairly suggests the foregoing additional recited feature of Claim 16.

For at least these reasons, Applicant respectfully submits that Claim 16, and claims that depend therefrom, are neither disclosed nor suggested by Leblang. Claim 37 recites features similar to those of Claim 16 as pointed out above which are neither disclosed nor suggested by Leblang. Thus, Claim 37, and claims that depend therefrom, are also neither disclosed nor suggested by Leblang for reasons similar to those regarding Claim 16.

In view of the foregoing, Applicant respectfully requests that the rejection be reconsidered and withdrawn.

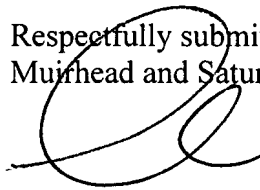
The rejection of Claims 11-15 and 32-36 under 35 U.S.C 103(a) as being unpatentable over Leblang is hereby traversed and reconsideration thereof is respectfully requested.

Claims 11-15 depend from Claim 1. Claims 32-36 depend from Claim 22. For reasons set forth above, Claims 1 and 22, and claims that depend therefrom are neither disclosed nor suggested by Leblang. Thus, Claims 11-15 and 32-36 are neither disclosed nor suggested by Leblang for at least the reasons set forth above regarding independent Claims 1 and 22.

In view of the foregoing, Applicant respectfully requests that the rejection be reconsidered and withdrawn.

Based on the above, Applicant respectfully requests that the Examiner reconsider and withdraw all outstanding rejections and objections. Favorable consideration and allowance are earnestly solicited. Should there be any questions after reviewing this paper, the Examiner is invited to contact the undersigned at 508-898-8604.

Respectfully submitted,  
Muirhead and Saturnelli, LLC



---

Anne E. Saturnelli  
Reg. No. 41,290

Muirhead and Saturnelli, LLC  
200 Friberg Parkway, Suite 1001  
Westborough, MA 01581  
Tel: (508) 898-8601  
Fax: (508) 898-8602

Date: April 19, 2007